Моделирование надежности программного обеспечения встраиваемых систем

Е. В. Душутина
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
elenvkov@gmail.com

Аннотация. **Обосновывается один из возможных подходов к оценке надежности программного обеспечения встраиваемых систем реального времени, уточняются особенности функционирования таких систем, критерии и показатели оценки. Известны многие фреймворки, нацеленные на измерение и прогнозирование надежности программных продуктов, в основе которых используются модели роста надежности программного обеспечения и неоднородного пуассоновского процесса. Однако очень немногие работы посвящены подходам к оценке программного обеспечения систем реального времени. Предлагается фреймворк для анализа применимости и точности моделей в зависимости от характера исходной информации и решаемых задач. Кратко рассматриваются наиболее подходящие модели надежности, вошедшие в состав предлагаемых средств, в соответствии с различным поведением функции риска во времени, сравниваются системы допущений этих моделей. Разработанный инструментарий позволяет выбирать подходящую модель или набор моделей для предсказания оставшихся в программном обеспечении дефектов. Исследования проводятся на основе результатов отладки и функционирования системы мониторинга состояния электрооборудования. Программное обеспечение такой системы реализует задачи смешанной критичности с определенным набором периодических и спорадических задач в режиме реального времени.**

Ключевые слова: встраиваемые системы реального времени, модели надежности программного обеспечения, SRGM

I. Введение

При разработке программного обеспечения (ПО) управляющих компьютерных систем исходят из динамических характеристик внешней среды и требований к временным характеристикам функционирования объекта управления и программ. При этом учитываются: инерционность объекта, являющегося источником или потребителем информации; реактивность системы или допустимая длительность ожидания отклика от момента поступления исходных данных до момента выдачи управляющего воздействия; периодичность решения задач по обработке информации для данного объекта. Программные отказы приводят к прекращению выдачи информации и управляющих воздействий или к значительному искажению содержания и темпа выдачи. В общем виде под отказом понимают нарушение работоспособности изделия и его соответствия требованиям технической документации [1]. Однако такое определение не является однозначным, если в качестве изделия рассматривать программу. Проявиться отказ может вследствие: искажения программного кода в памяти команд, искажения данных в оперативной или долговременной памяти, нарушения нормального хода вычислительного процесса под управлением операционной системы, несовершенства планирования задач, приводящего к нарушению сроков выполнения заданий и т.д. Каждое из этих нарушений может быть как результатом отказа или сбоя аппаратуры, так и ПО. Если в качестве критерия надежности программы рассматривать ее способность обеспечить низкую вероятность отказа в процессе функционирования посредством восстановления (или нивелирования) искаженной информации и вычислительного процесса за время, не превышающее некоторое пороговое значение [1, 2], то основными показателями надежности при таком подходе, как правило, выбираются: наработка на отказ (т. к. система считается восстанавливаемой) и коэффициент доступности [3]. Последний соответствует доле времени полезной работы программ на достаточно большом интервале, содержащем отказы и восстановления. Целесообразнее рассматривать подобные программы как программы, содержащие в своем составе дополнительные средства для повышения надежности компьютерной системы в целом за счет избыточного программного кода, контрольных точек, специальных механизмов организации вычислительного процесса, позволяющих своевременно восстановить работоспособность системы [3, 4]. В этом случае не учитывается природа и причина возникновения отказов и сбоев, одинаково оценивается нарушение работоспособности программ как в результате ошибки в самой программе, так и в результате аппаратного дефекта или иных внешних причин.

Таким образом, можно разделить собственно оценку надежности программ и целостную оценку программно-аппаратного комплекса или системы. При этом один из подходов к оценке надежности программного кода – это учет и прогнозирование оставшихся в ПО ошибок посредством анализа функции риска на основе SGRM.

A. Общая постановка задачи исследования

Оба аспекта (оценка собственно кода и оценка надежности системы в целом) не противоречат друг другу, а скорее дополняют одно другое. Поэтому в исследовании рассматривается комплексный подход, который предполагает:

- разработку избыточного ПО с целью повышения готовности (доступности) управляющей системы в любой момент выполнять заданные функции, т. е. повышения ее отказоустойчивости за счет программной избыточности;

- оценку надежности системы до внедрения в систему избыточных программных средств;
- оценку надежности программного кода и прогнозирование оставшихся ошибок в коде исходной системы;
- оценку надежности программного кода и прогнозирование оставшихся ошибок в дополненном коде избыточными средствами повышения готовности;
- оценку надежности после внедрения избыточных средств системы в целом и сравнение с надежностью исходной системы.

В данной статье внимание уделяется оценке надежности программного кода с учетом особенностей входных потоков и функций риска, наиболее подходящих для рассматриваемого типа систем. С этой целью предлагается фреймворк для анализа применимости и точности моделей в зависимости от характера исходной информации и решаемых задач. Фреймворк включает расширяемый набор моделей, методов и инструментальные программные средства для их реализации с целью упрощения и частичной автоматизации процесса оценки.

B. Краткий обзор предметной области

За последние несколько десятилетий были накоплены обширные знания и практический опыт применения моделей надежности ПО. Особое внимание уделяется моделям роста надежности программного обеспечения. Надежность программных систем оценивается с помощью различных подходов, включая Марковские модели [5], регрессионный анализ [6,12] программные инварианты [7]. В последнее время активно развиваются методы на основе нечетких моделей [8, 9, 10], нейронных сетей [11, 12, 13], гранулярных моделей [14], методы машинного обучения [11, 15]. Интерес к моделям роста надежности программного обеспечения не ослабевает, появляются их многочисленные расширения и модификации. В [16] предлагается расширение непараметрических SRGM, использующих регрессию опорных векторов для прогнозирования вероятностных показателей времени до отказа.

Справедливости ради следует сказать, что в ряде статей утверждается, что модели надежности ПО не завоевывают доверия в промышленном сообществе [17]. С точки зрения разработчика надежность ПО, в первую очередь, связана с дефектами, которые представляют собой основной фактор затрат при разработке. Это называется подходом, ориентированным на разработчиков [18]. В этом контексте требуются более реалистичные и точные показатели надежности ПО. Разрабатывается много фреймворков и моделей, учитывающих дополнительные факторы, такие как метрики программ [19], качество отладочного процесса [20], квалификация программиста и т.д.

II. Модели и методы исследования

Для оценки надежности будем использовать показатели:

- число оставшихся в программе ошибок B – число ошибок, которые могут быть обнаружены на следующих этапах разработки ПО по отношению к рассматриваемому этапу после внесения исправлений на этом этапе;
- вероятность безотказной работы ПО *P(t)* – на заданном интервале времени [0; *t*], а также статистическая оценка вероятности отказа;
- наработка ПО до отказа $t_{cp} = \int_0^\infty P(t)\,dt$;
- наработка на отказ для систем с восстановлением
- *функция риска* или интенсивность обнаружения ошибок (что равносильно интенсивности отказов).

Вероятность хотя бы одного отказа за время *t* есть $Q(t) = 1 - P(t)$, тогда плотность вероятности отказа $q(t) = dQ(t)/dt = -dP(t)/dt$.

Статистическая оценка вероятности отказа за *m* независимых прогонов программы

$$\hat{Q}(m) = \frac{1}{m}\sum_{i=1}^{m} \varphi_i,$$

где φ_i – дихотомическая переменная, принимающая значения 1 или 0 в зависимости от наличия или отсутствия отказа соответственно. При этом отказом считается превышение допустимого предельного значения результата, полученного при i-ом прогоне программы на некотором i-ом наборе входных данных.

Пусть функция риска *R(t)* – условная плотность вероятности отказа ПО (с размерностью [1/время]) в некоторый момент времени *t* при условии, что до этого момента отказа еще не произошло:

$$R(t) = q(t)/P(t) = -[1/P(t)]dP(t)/dt \text{ или}$$
$$dP(t)/P(t) = -R(t)dt$$
$$\Rightarrow \ln P(t) = -\int_0^t R(t)dt,$$

т. е. $P(t) = exp\{-\int_0^t R(t)dt\}$

Различное поведение функции риска во времени является основанием для рассмотрения множества различных моделей надежности ПО. Сформулируем общий набор признаков, пригодный для формирования системы допущений, соответствующей той или иной модели. Для всех моделей полагаем, что ПО функционирует в среде близкой к реальным условиям, а также, что ошибки постоянно корректируются без внесения новых. Для каждой модели необходимо определить:

- вероятность каждой из ошибок и корреляцию появления ошибок (1);
- веса или значимость различных типов ошибок (2);
- распределение времени до следующего отказа (3);
- характер изменения функции риска в рассматриваемом интервале (например, между двумя смежными моментами появления ошибок в классических моделях) (4);
- характер изменения функции риска от интервала к интервалу: на всем интервале тестирования (5);
- допустимое количество ошибок в рассматриваемом интервале (6);

- количество корректируемых ошибок n_i из обнаруженных на данном временном интервале f_i , если допустимое количество ошибок в интервале больше единицы (7);

Включенные в состав предлагаемого фреймворка модели на основе функции риска приведены в таблицах 1 и 2 [1, 21, 22]. По вертикали указаны названия моделей, а по горизонтали – основные допущения (пронумерованы выше), принятые в каждой модели. В табл. 1 сгруппированы базовые модели с допущениями о равной вероятности, независимости и равновесности ошибок. В табл. 2 приведены геометрические модели.

ТАБЛИЦА I. БАЗОВЫЕ МОДЕЛИ С ДОПУЩЕНИЯМИ (1 – 2) О РАВНОЙ ВЕРОЯТНОСТИ, НЕЗАВИСИМОСТИ И РАВНОВЕСНОСТИ ОШИБОК

	3	4	5 ($R(t)$)	6 (7)
Джелинского–Моранды	exp	$const$	Kb_i	≤ 1
Простая экспоненциальная	exp	$f(N(t))$	$KB\exp\{-Kt\}$	N(t)
Шика–Уолвертона	exp	$f(b_i, t)$	Kb_iT_i	≤ 1
Липова (обобщение моделей 1 и 3)	t_i= fix	$const$	$K[B - F_{i-1}]$	f_i, $n_i \leq f_i$
Липова (обобщение модели 3)	t_i=fix	$K(B-F_{i-1})(\sum_{j=1}^{i-1} T_j + \frac{T_j}{2})$		f_i
Модель Дюэна	t– случ.	$B(t)$ распределено по закону Пуассона		–

ТАБЛИЦА II. ГЕОМЕТРИЧЕСКИЕ МОДЕЛИ (МОДИФИКАЦИИ БАЗОВЫХ)

Геометрическая модель (модификация модели 1 Джелинского -Моранды)	неравно вероятные ошибки, из число (B) не фиксированное	exp	$const$, геометр. прогрес-сия	DK^{i-1}, $(0 < K < 1)$	–
Геометрическая модель Липова	неравно вероятные ошибки, более слабое условие неограниченности B	exp	$R(t) = DK^{n_{i-1}}$,где (0<K<1), n_{i-1}– число ошибок за все интервалы тестирования, ∀ ошибка исправляется или не учитывается		f_i,
Геометрическая модель Моранды	неравно вероятные ошибки, число ошибок B неограниченно	t_i=fix	Закон Пуассона	$D\Phi^{i-1}$, $(0 < \Phi < 1)$	f_i

Приведем некоторые замечания, которые не нашли отражения в таблице, для других моделей. В *простой экспоненциальной* модели (2) снято допущение о постоянстве функции риска в интервале между смежными моментами появления ошибок по сравнению с первой моделью (1). $N(t)$ – число обнаруженных к моменту t ошибок, причем $R(t)=dN(t)/dt$. В модели (3) *Шика–Уолвертона* функция риска $R(t)$ пропорциональна, и текущему числу оставшихся ошибок, и времени тестирования. Кроме того, предполагается, что по мере тестирования программы возрастают шансы обнаружения последующих ошибок. В *первой* модели *Липова* (4) на i-м интервале тестирования корректируется лишь часть n_i из обнаруживаемых ошибок f_i, причем число отказов f_i – случайная величина с распределением Пуассона, $F_{i-1} = \sum_{j=1}^{i-1} n_j$ – общее число скорректированных к моменту t_{i-1} ошибок. В основе *второй* модели *Липова* (5) предполагается, что уровень обнаружения ошибок пропорционален текущему числу ошибок, среднему времени поиска ошибки из текущего интервала, а также общему времени тестирования до начала текущего интервала.

Геометрические модели получили свое название, из-за допущения, что интенсивность обнаружения ошибок образует геометрическую прогрессию со знаменателем K ($0<K<1$) и исходной интенсивностью D. Причем, в первых двух моделях в интервале между появлениями ошибок интенсивность постоянна, а в третьей – число обнаруженных ошибок подчиняется закону Пуассона с параметром $D\Phi^{i-1}$, где Φ – коэффициент пропорциональности ($0<\Phi<1$). Кроме того, в третьей модели считается, что каждая обнаруженная ошибка или исправляется или повторно не учитывается. Эта модель применима, когда длина интервала мала по сравнению со всем временем тестирования. В *первой* модели полагается, что по мере отладки обнаруживать ошибки становится сложнее. Геометрические модели удобны для оценки средней наработки до отказа и вероятности безотказной работы. Для оценок параметров моделей D, K, Φ используется метод максимального правдоподобия.

В модели *Дюэна* рассматривается отношение интенсивности обнаружения ошибок к общему времени тестирования. При этом общее число ошибок $B(t)$, обнаруженных к произвольному моменту времени t, распределено по закону Пуассона со средним значением $m(t)$. Исследуемая при этом функция: $m(t)/t$. Для реализации этой модели требуется только знание моментов времени t_i появления ошибок.

В таблицу не вошли модели Шнейдевинда и Вейбулла [22]. В модели Вейбулла плотность распределения отказов (время до появления очередной ошибки) описывается распределением Вейбулла. Функция риска описывается в виде

$$R(t) = \left(\frac{a}{b}\right)\left(\frac{t}{b}\right)^{a-1},$$

где $a > 0$, $b > 0$ – константы модели, $t \geq 0$ (интервал безошибочной работы). Если $a > 1$, интенсивность обнаружения со временем растет, если $a < 1$ – падает, при $a = 1$ функция постоянна. Для получения оценок используется метод наименьших квадратов (МНК). Метод удобен для оценки средней наработки до отказа.

В модели Шнейдевинда предполагается, что на процесс прогнозирования более сильное влияние оказывают появления более поздних ошибок. Для этого из m интервалов тестирования рассматриваются: 1) либо данные об обнаруженных ошибках во всех m интервалах для предсказания будущего состояния ПО; 2) либо данные последних (m-s+1) интервалов для предсказания в случаях, когда в процессе обнаружения произошло существенное изменение в интервале s; 3) суммарные данные с первого по (s-1)-й интервалы, включая особые ошибки в последних с s-го по m-й интервал. Процесс обнаружения ошибок предполагается неоднородным процессом Пуассона с экспоненциально уменьшающимся уровнем обнаружения ошибок. Основные допущения при этом следующие: 1) число ошибок в данном интервале не зависит от числа ошибок в других интервалах; 2) число обнаруженных ошибок уменьшается от интервала к интервалу; 3) все интервалы

имеют одинаковую длину; 4) интенсивность обнаружения пропорциональна текущему числу ошибок. Третья геометрическая модель Моранды является частным случаем модели Шнейдевинда.

III. Результаты: применение моделей роста надежности программного обеспечения

Для оценки надежности программных комплексов и прогнозирования оставшихся ошибок был разработан программный фреймворк, основные функции которого:

- сбор информации на этапах отладки и сопровождения ПО;
- автоматизированная оценка сложности испытуемой программы;
- обработка собранной или сгенерированной отладочной информации посредством применения адаптированных моделей на основе функции риска (в том числе приведенных выше) и иных моделей.

С помощью моделей риска анализировались прикладные программы, написанные на языках ассемблера и C/C++, предназначенные для функционирования в реальном времени под управлением целевой операционной системы реального времени. Для анализа использовались данные отладочной отчетности и моделирования (искусственной генерации ошибок с заданными распределениями и параметрами моделей надежности ПО).

Первые версии (релизы) фреймворка использовались в проектах «Сбор информации о состоянии распределенных объектов» и «Анализ состояния электрооборудования» № государственной регистрации программы для ЭВМ 2013610021 и №2013611687 [23] соответственно, далее они получили свое развитие. Кроме того, количественные оценки показателей надежности исследовались применительно к программным средствам обеспечения отказоустойчивости, встраиваемым в микроядерную операционную систему реального времени управляющего вычислительного комплекса.

Система показателей надежности была модифицирована в соответствии с требованиями, предъявляемыми к ПО систем управления (рассматриваемых как восстанавливаемые системы). В частности, для оценки ПО систем управления были введены следующие показатели:

- отладочная вероятность безотказной работы ПО, характеризующая надежность ПО по окончании его комплексной отладки и испытаний на стенде, объекте в составе системы;
- расчетное время сопровождения ПО (время необходимой наработки ПО на начальном этапе эксплуатации системы);
- эксплуатационные показатели надежности, учитывающие этап сопровождения ПО, – эксплуатационная вероятность безотказной работы за заданное время непрерывной работы и/или коэффициент готовности (доступности) ПО (в зависимости от временных режимов его работы);
- допустимое время восстановления.

Для длительно решаемых прикладных задач рассчитывается эксплуатационная вероятность или функция риска за заданный период непрерывного функционирования. Для эпизодически решаемых задач рассчитывается эксплуатационный коэффициент готовности ПО.

Для исследования использовались данные результатов отладки и функционирования системы мониторинга состояния электрооборудования. Детальное описание экспериментальной системы представлено в [24], в ее состав входит широкий спектр датчиков, устанавливаемых непосредственно на контролируемых устройствах и вблизи них, в различных точках рабочего помещения. Управление осуществляется встроенной интеллектуальной системой онлайн-мониторинга. Выполняются параметрические корректировки для расчета средних и предельных диапазонов. Предусмотрена управляющая реакция, когда значения параметров выходят за пределы допустимого диапазона. Передача данных из встроенной подсистемы на более высокий уровень всей системы осуществляется для архивации, документирования и хранения данных. Таким образом, осуществляется полный цикл обработки данных с выполнением функциональных задач разного типа. При этом задачи для системы мониторинга весьма разнообразны и сильно отличаются друг от друга по способам выполнения, вычислительной нагрузке, характеру поступления задач и степени их периодичности, срокам выполнения и т. п. Для их своевременного выполнения применяются методы статического планирования для набора задач жесткого реального времени высокого уровня критичности и динамического планирования (EDF) для периодических и спорадических задач.

На основе проведенных экспериментов по анализу рисков и прогнозированию были сделаны следующие выводы общего характера.

Главная проблема практически всех перечисленных моделей – достижение сходимости процедур оценки параметров моделей: это может быть расходящийся или не сходящийся процесс, или сходящийся, но к неоптимальному решению, осцилляция, неоднозначность решения. Улучшению свойств сходимости на практике способствует точное соблюдение допущений, предусмотренных в модели, и выявление реального распределения ошибок во времени. Ухудшение сходимости наблюдается при увеличении длины тестового интервала.

Модель функции риска выбирается в зависимости от имеющихся исходных данных. При наличии информации о длительности интервалов между появлениями ошибок целесообразно применять геометрические модели, тогда как при известном количестве ошибок целесообразно воспользоваться моделями Шнейдевинда. Следует отметить, что входные потоки данных ожидаемо лучше согласуются с пуассоновскими распределениями. В моделях на основе метода максимального правдоподобия точность оценок возрастает, если увеличивается общее количество ошибок или уменьшается число оставшихся в программе ошибок. Прогнозируемое число оставшихся в ПО ошибок в моделях с экспоненциальными

распределениями при прочих равных условиях систематически меньше, чем в других моделях, что свидетельствует о тенденции некоторого искажения результатов (в сторону улучшения прогноза) при анализе функций риска экспоненциального характера.

IV. Заключение

Несмотря на то, что на сегодняшний день предложено огромное количество моделей, до сих пор не существует универсального подхода к построению адекватной модели надежности, который легко можно было бы реализовать. Предлагаемые в этой работе средства помогают решать эту проблему для управляющих встраиваемых систем определенного класса.

Применение предложенного фреймворка позволяет выполнить этапы исследования, связанные с оценкой надежности программного кода и прогнозирования оставшихся ошибок в коде исходной системы; а также кода избыточных программных средств, дополнительно встраиваемых в систему с целью повышения ее отказоустойчивости. Дальнейшие исследования целесообразно направить на разработку подхода к оценке надежности системы в целом до и после внедрения избыточных программных средств и на их сравнение.

Список литературы

[1] Israel Koren, C. Mani Krishna. Fault tolerant systems. Morgan Kaufmann Publishers. 378, (2007).

[2] Lyu M.R. (Ed.). Handbook of Software Reliability Engineering. McGraw-Hill, Inc., Hightstown, NJ, USA. 819, (1996).

[3] Trivedi K.S., "Reliability and Availability Assessment in Practice," 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Cosenza, Italy, 2019, pp. 1-1, doi: 10.1109/DS-RT47707.2019.8958676. https://doi.org/10.1109/DS-RT47707.2019.8958676

[4] Trivedi K., & Bobbio A. (2017). Reliability and Availability Engineering: Modeling, Analysis, and Applications. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781316163047.

[5] Bharathi R., Selvarani R. Hidden Markov Model Approach for Software Reliability Estimation with Logic Error. In: Int. J. Autom. Comput. 17, 305–320 (2020). https://doi.org/10.1007/s11633-019-1214-7.

[6] Matsumoto E.Y., Del-Moral-Hernandez E. Improving regression predictions using individual point reliability estimates based on critical error scenarios. In: Inf. Sci. (Ny) 374 (2016) 65–84.

[7] Z. Ding, M.-H. Chen, X. Li, Online reliability computing of composite services based on program invariants, Inf. Sci. (Ny) 264 (2014) 340–348

[8] Diwaker C., Tomar P., Poonia R.C. et al. Prediction of Software Reliability using Bio In-spired Soft Computing Techniques. J Med Syst 42, 93 (2018). https://doi.org/10.1007/s10916-018-0952-3

[9] Diwaker C. et al., "A New Model for Predicting Component-Based Software Reliability Using Soft Computing," in IEEE Access, vol. 7, pp. 147191-147203, 2019, doi: 10.1109/ACCESS.2019.2946862.

[10] Garg H., "A Brief Analysis of Soft Computing Techniques in Software Fault Prediction," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-7, doi: 10.1109/ISCON52037.2021.9702418.

[11] Jindal A., Gupta A. and Rahul, "Comparative Analysis of Software Reliability Prediction Using Machine Learning and Deep Learning," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 389-394, doi: 10.1109/ICAIS53314.2022.9743129.

[12] Yakovyna V., Uhrynovskyi B. and Bachkay O., "Software Failures Forecasting by Holt - Winters, ARIMA and NNAR Methods," 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 151-155, doi: 10.1109/STC-CSIT.2019.8929863. https://doi.org/10.1109/STC-CSIT.2019.8929863

[13] C. De Sio, S. Azimi and L. Sterpone, "FireNN: Neural Networks Reliability Evaluation on Hybrid Platforms," in IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 2, pp. 549-563, 1 April-June 2022, doi: 10.1109/TETC.2022.3152668. https://doi.org/10.1109/TETC.2022.3152668

[14] Kiersztyn A., Karczmarek P., Kiersztyn K. and Pedrycz W., "The Concept of Detecting and Classifying Anomalies in Large Data Sets on a Basis of Information Granules," 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 2020, pp. 1-7, doi: 10.1109/FUZZ48607.2020.9177668. https://doi.org/10.1109/FUZZ48607.2020.9177668

[15] Jagtap M., Katragadda P. and Satelkar P., "Software Reliability: Development of Software Defect Prediction Models Using Advanced Techniques," 2022 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, USA, 2022, pp. 1-7, doi: 10.1109/RAMS51457.2022.9893986. https://doi.org/10.1109/RAMS51457.2022.9893986

[16] Utkin L.V., Coolen F.P.A. A robust weighted SVR-based software reliability growth model. In: Reliability engineering and system safety, 176. 2018, pp.93-101. https://doi.org/10.1016/j.ress.2018.04.007 .

[17] Sanjay L. Joshi, Bharat Deshpande, Sasikumar Punnekkat, Do Software Reliability Predic-tion Models Meet Industrial Perceptions. In: ISEC '17 Proceedings of the 10th Innovations in Software Engineering Conference, 2017, pp. 66-73.

[18] Ivanov V., Rogers A., Succi G., Yi J., and Zorin V. What do software engineers care about? Gaps between research and practice. In: Proceedings of the 11th Joint Meeting on Foundations of Software Engineering. ACM, 2017, pp. 890–895.

[19] Ivanov V., Reznik A., Succi G., Comparing the reliability of software systems: A case study on mobile operating systems. In: Information Sciences v. 423, 2018, pp. 398-411, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2017.08.079 .

[20] Cinque M, Cotroneo D, Pecchia A, Pietrantuono R, Russo S. Debugging-workflow-aware software reliability growth analysis. Softw Test Verif Reliab. 2017; 27:e1638. https://doi.org/10.1002/stvr.1638

[21] Jelinski Z., Moranda P.B. (1972). Software reliability research. In: W. Greiberger, editor, Statistical Computer Performance Evaluation, pp 464–484. Academic Press, New York.

[22] David D. Hanagal, Nileema N. Bhalerao, Software Reliability Growth Models. Infosys Science Foundation Series in Mathematical Sciences. Springer Singapore. 2021, p.125. https://doi.org/10.1007/978-981-16-0025-8

[23] Душутина Е.В. Сбор информации о состоянии распределенных объектов. Свидетельство РФ о государственной регистрации программ № 2013610021 (2013).

[24] Душутина Е.В. Анализ состояния электрооорудования. Свидетельство РФ о государственной регистрации программ №2013611687 (2013).

[25] Dushutina E.V. Mixed-criticality Application Scheduling in Safety-involved Embedded Systems, 2022 XXV International Conference on Soft Computing and Measurements (SCM), SPb, RF, 2022, pp. 244-249, doi: 10.1109/SCM55405.2022.9794839.